

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problems Mailbox.**

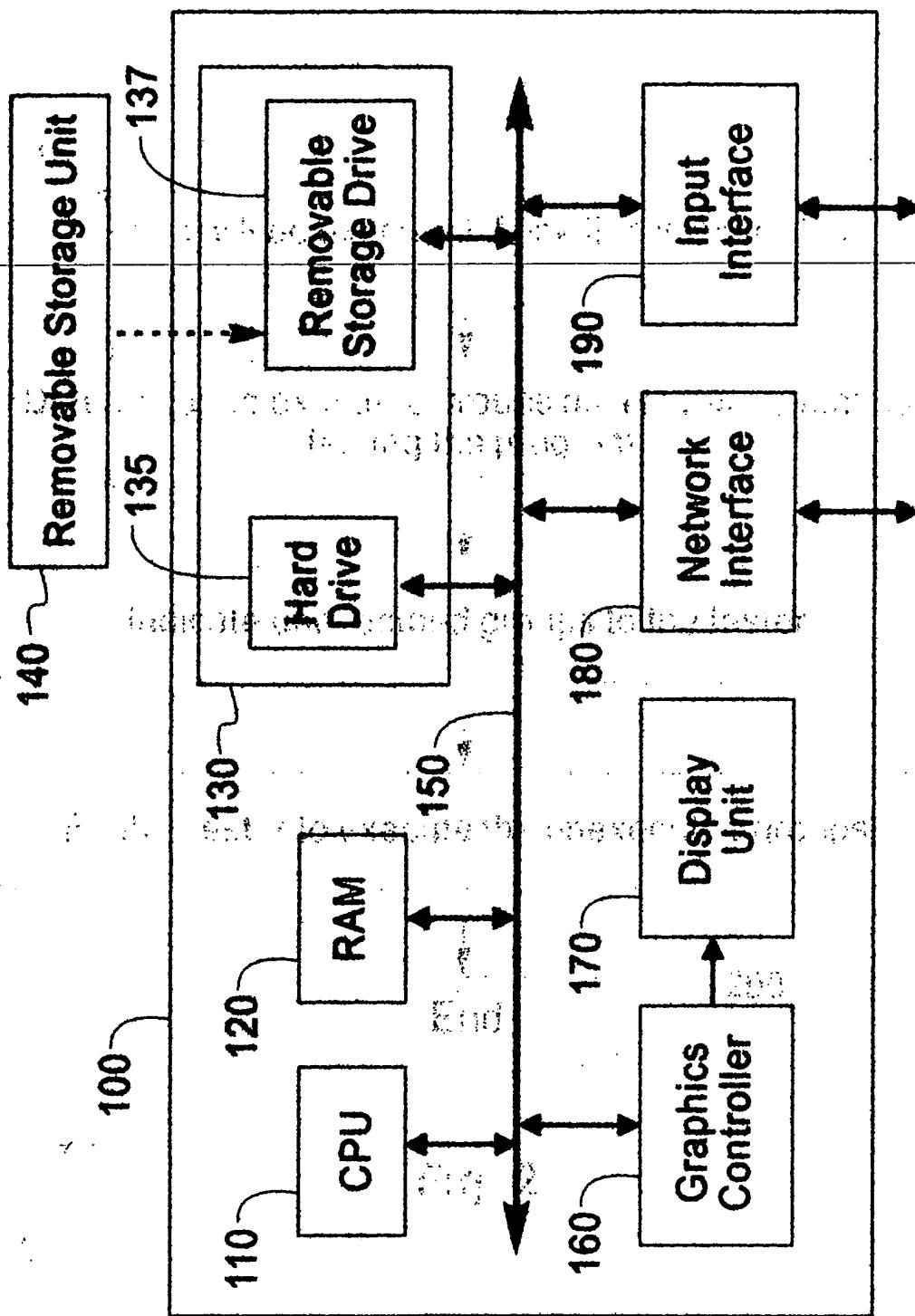


Fig. 1

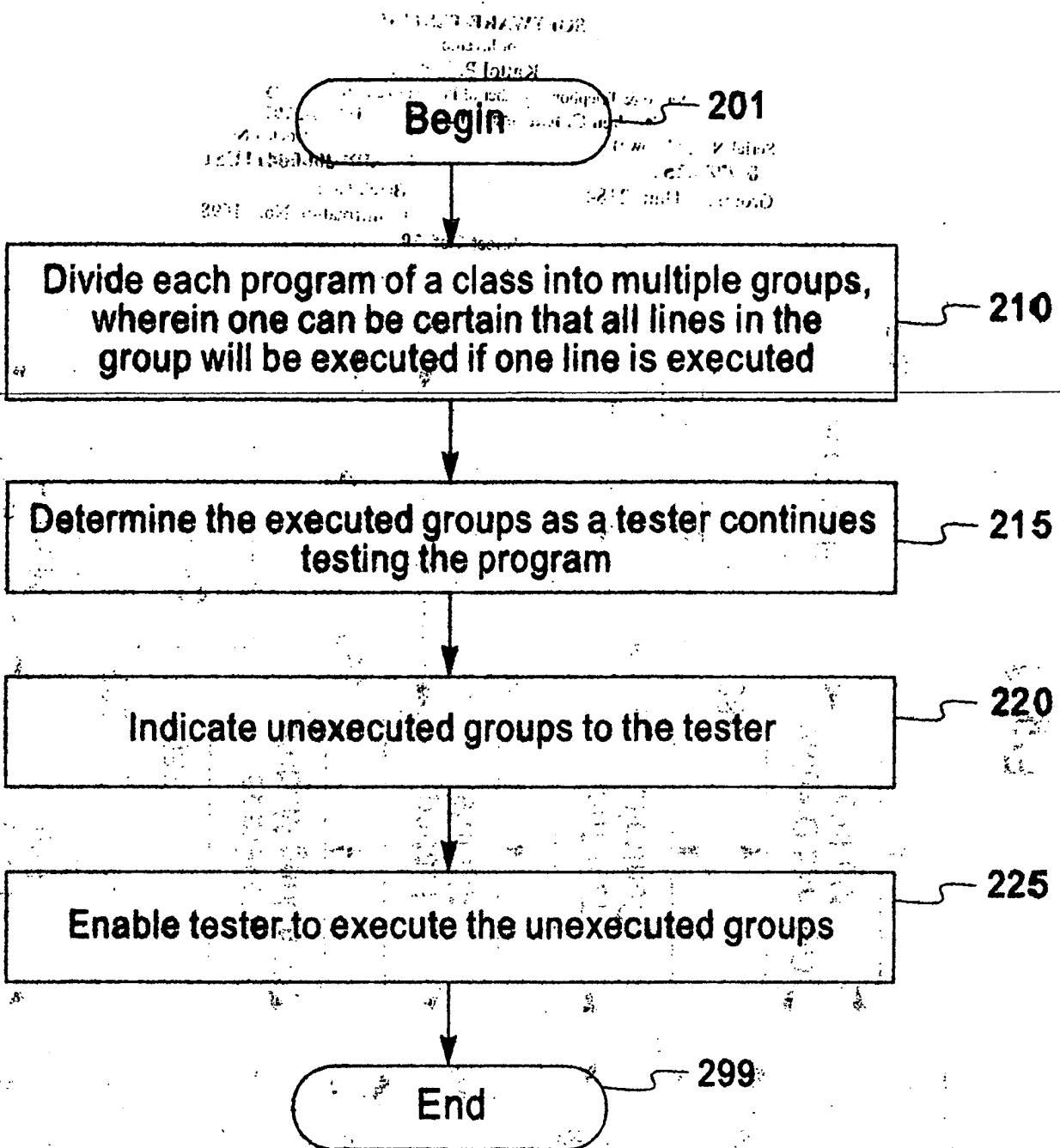


Fig. 2

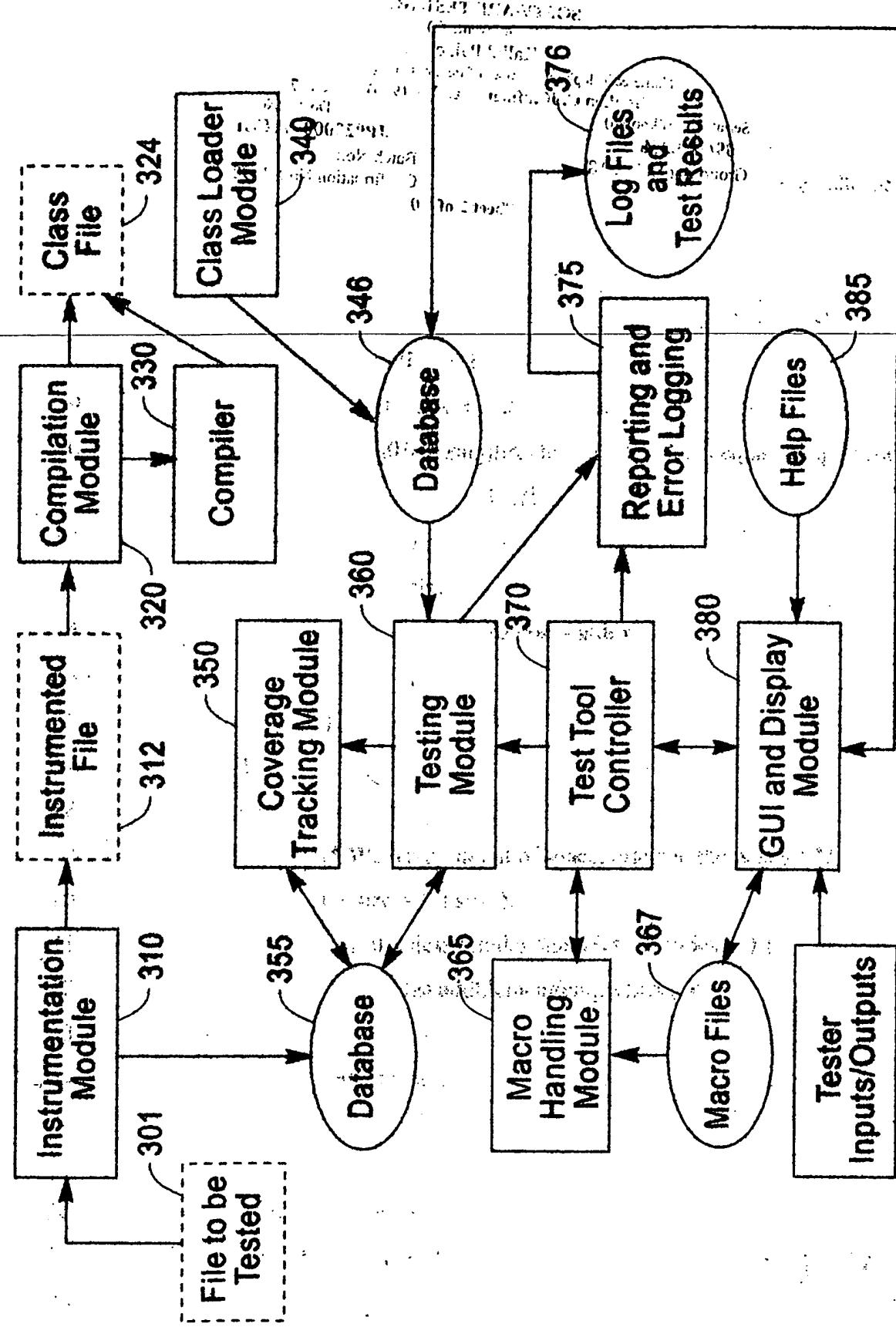


Fig. 3

```

1 public void addString(String inStr) {
2     /* The location where the string needs to be set */
3     int setPos = -1;
4     /* Check the size to see if the next set needs to be allocated */
5     if(list.size() > 0 && pos ==
6         (String)(list.elementAt(list.size()-1))).equals(emptyString)) {
7         /* We still have empty space, get the first empty slot */
8         int totalRec = list.size();
9         for(int index=totalRec-1;index >= 0;index--) {
10             if(!((String)list.elementAt(index)).equals(emptyString)) {
11                 setPos = index;
12                 break;
13             } else {
14                 setPos = index;
15             }
16         }
17     }
18     else {
19         /* We have run out of space, allocate the next set */
20         int size = list.size();
21         for(int index=0;index<numSetVal;index++) {
22             list.addElement(emptyString);
23         }
24         setPos = size;
25     }
26     /* Add the element */
27     list.setElementAt(inStr,setPos);
28 }

```

Fig. 4A

```

1 public void addString(String inStr) {
2     /* INSTRUMENTATION Group B11 -BEGIN */
3     Wiz_Tracer.trackExecution("WizStringList",11);
4     /* INSTRUMENTATION GROUP B11 - END */
5     int setPos = -1;
6     if( list.size() > 0 && (
7         (String)(list.elementAt(list.size()-1))).equals(emptyString) )
8     {/*INSTRUMENTATION GROUP B12 - BEGIN */
9         Wiz_Tracer.trackExecution("WizStringList",12);
10        /* INSTRUMENTATION GROUP B12 - END */
11        int totalRec = list.size();
12        for(int index=totalRec-1;index >= 0;index--) {
13            /* INSTRUMENTATION GROUP B13 - BEGIN */
14            Wiz_Tracer.trackExecution("WizStringList",13);
15            /* INSTRUMENTATION GROUP B13 - END */
16            if(!((String)list.elementAt(index)).equals(emptyString)) {
17                /* INSTRUMENTATION GROUP B14 - BEGIN */
18                Wiz_Tracer.trackExecution("WizStringList",14);
19                /* INSTRUMENTATION GROUP B14 - END */
20                break;
21            } /* <if GROUP(B14) ends at line : 134> */
}

```

Fig. 4B

```

22     else { /* <else GROUP(B15) begins at line : 135> */
23         /* INSTRUMENTATION GROUP B15 - BEGIN */
24         Wiz_Tracer.trackExecution("WizStringList",15);
25         /* INSTRUMENTATION GROUP B15 - END */
26         setPos = index;
27     } /* <else block(B15) ends at line : 138> */
28     /* <break(B16) begins at line : 139> */
29     /* INSTRUMENTATION GROUP B16 - BEGIN */
30     Wiz_Tracer.trackExecution("WizStringList",16);
31     /* INSTRUMENTATION GROUP B16 - END */
32 } /* <for block(B13) ends at line : 139> */
33 } /* <if block(B12) ends at line : 141> */
34 else { /* <else block(B17) begins at line : 142> */
35     /* INSTRUMENTATION GROUP B17 - BEGIN */
36     Wiz_Tracer.trackExecution("WizStringList",17);
37     /* INSTRUMENTATION GROUP B17 - END */
38     int size = list.size();
39     for(int index=0;index<numSetVal;index++) { /* <for
40         /* <for block(B18) begins at line : 146> */
41         /* INSTRUMENTATION GROUP B18 - BEGIN */
42         Wiz_Tracer.trackExecution("WizStringList",18);
43         /* INSTRUMENTATION GROUP B18 - END */
44         list.addElement(emptyString);
45     } /* <for block(B18) ends at line : 149> */
46     setPos = size;
47 } /* <else block(B17) ends at line : 152> */
48 list.setElementAt(inStr,setPos);
49 } /* <method block - addString(B11) ends at line : 156> */

```

Fig. 4C

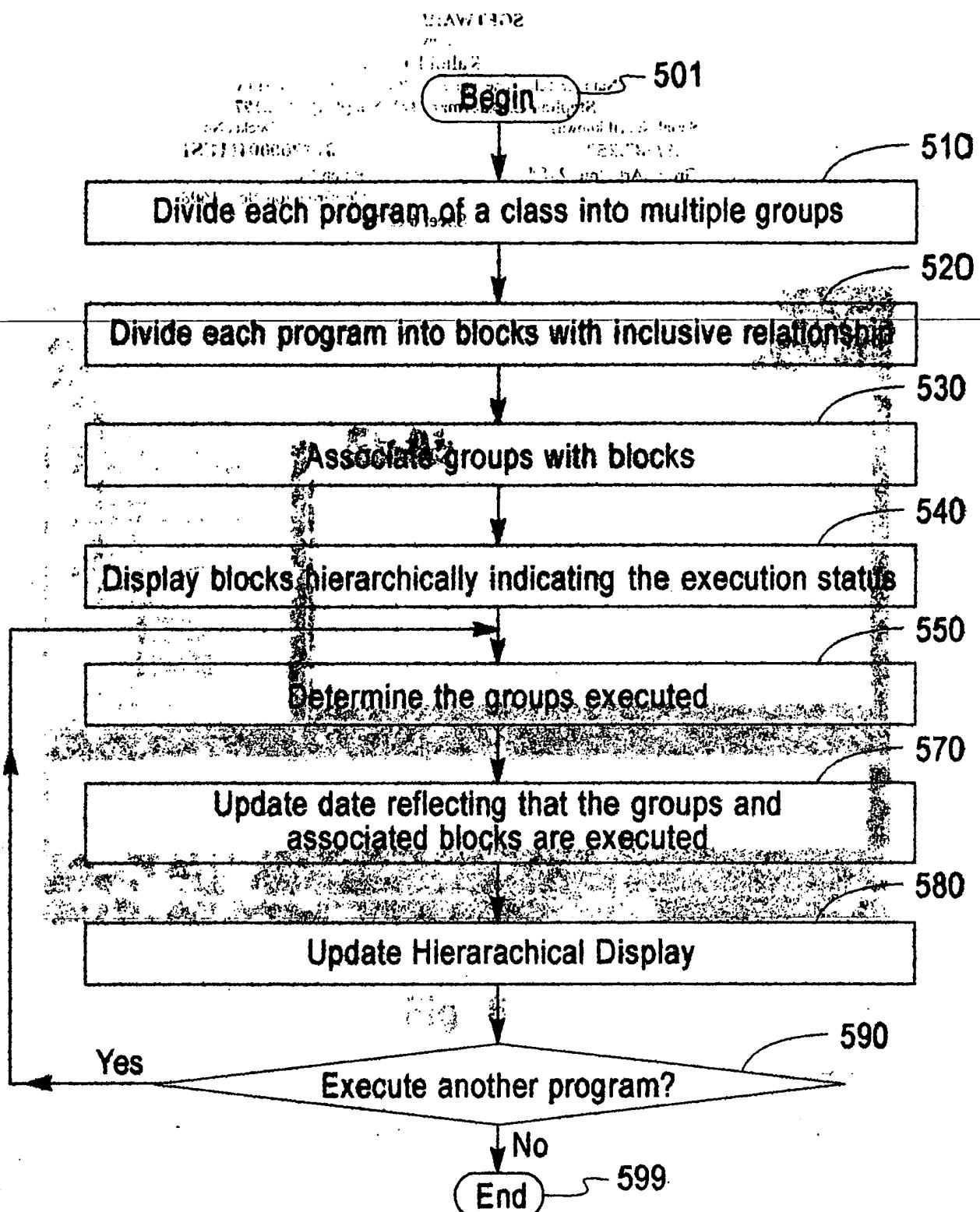


Fig. 5

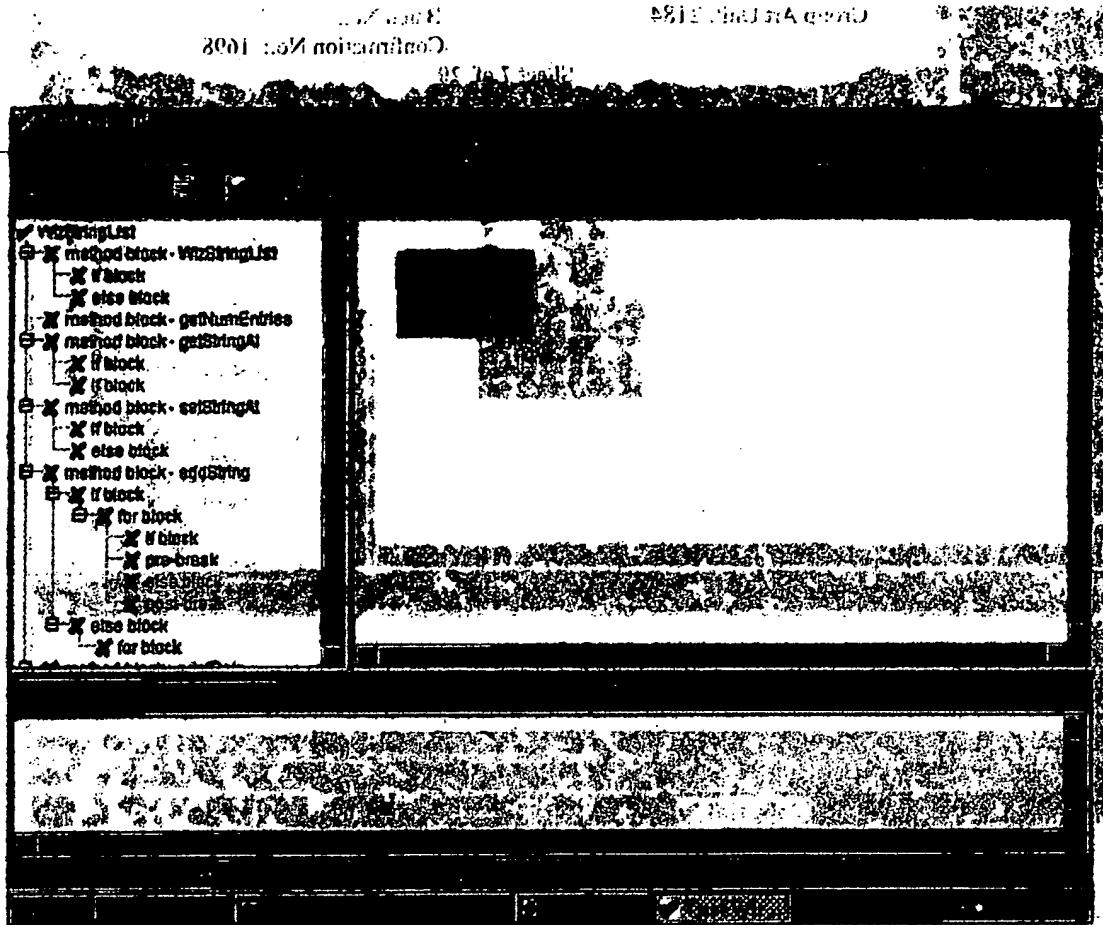
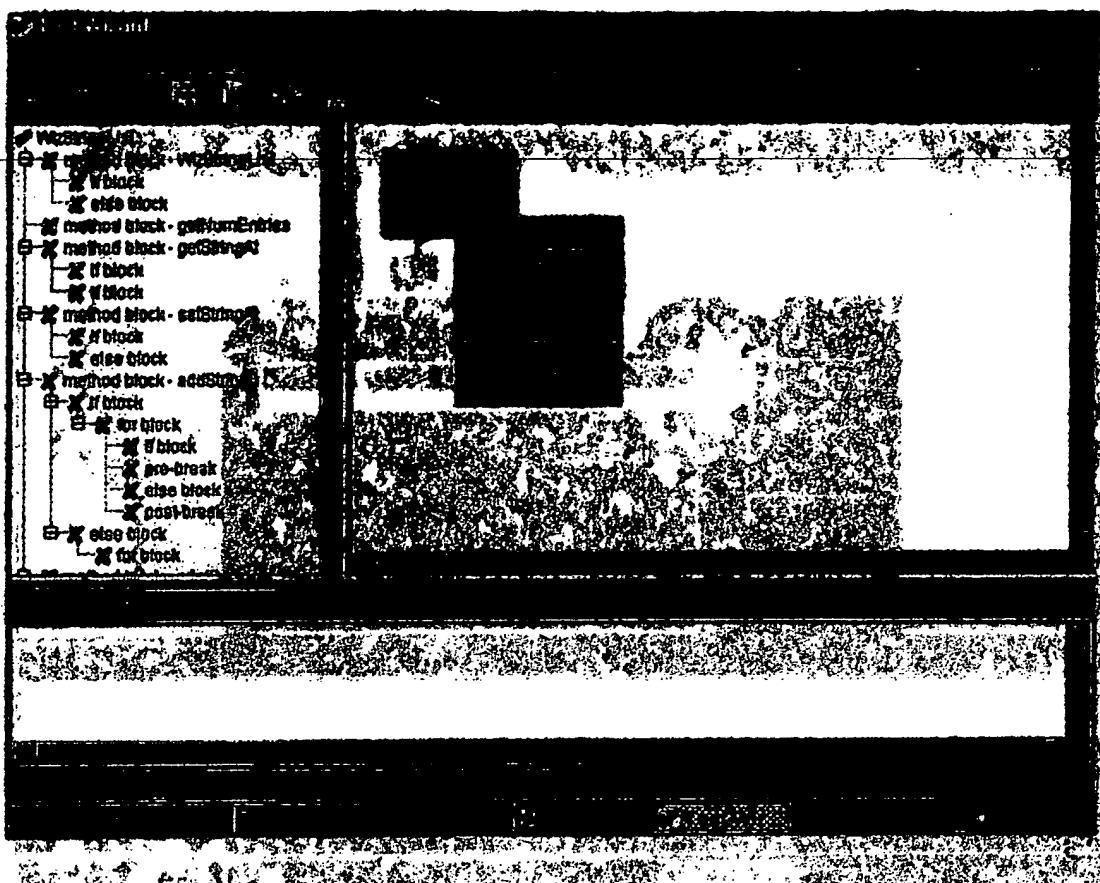


Fig. 6

9/20

K. Pal et al.  
S.C.K. JP920000411



**Fig. 7A**

Fig. 3

10/20  
K. Pal et al.  
S.C.K. JP920000411

--

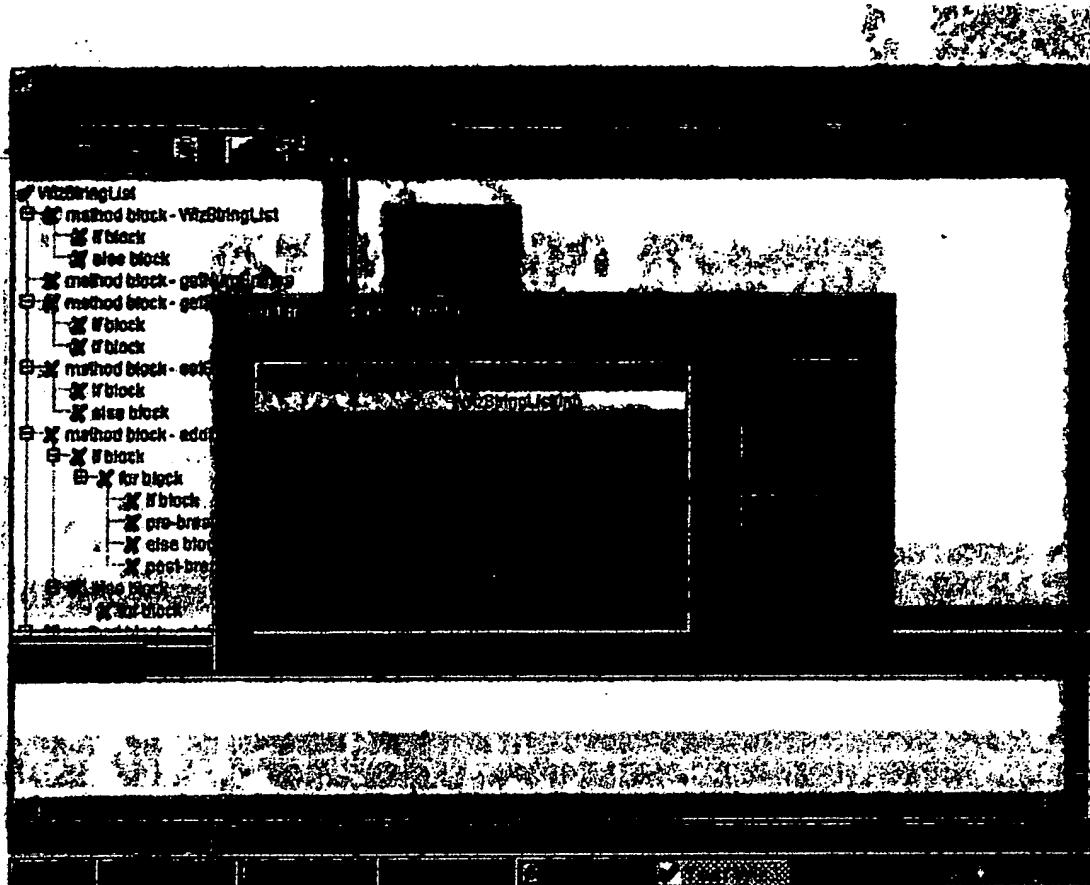


Fig. 20.

**Fig. 7B**

11/20

K. Pal et al.  
S.C.K. JP920000411

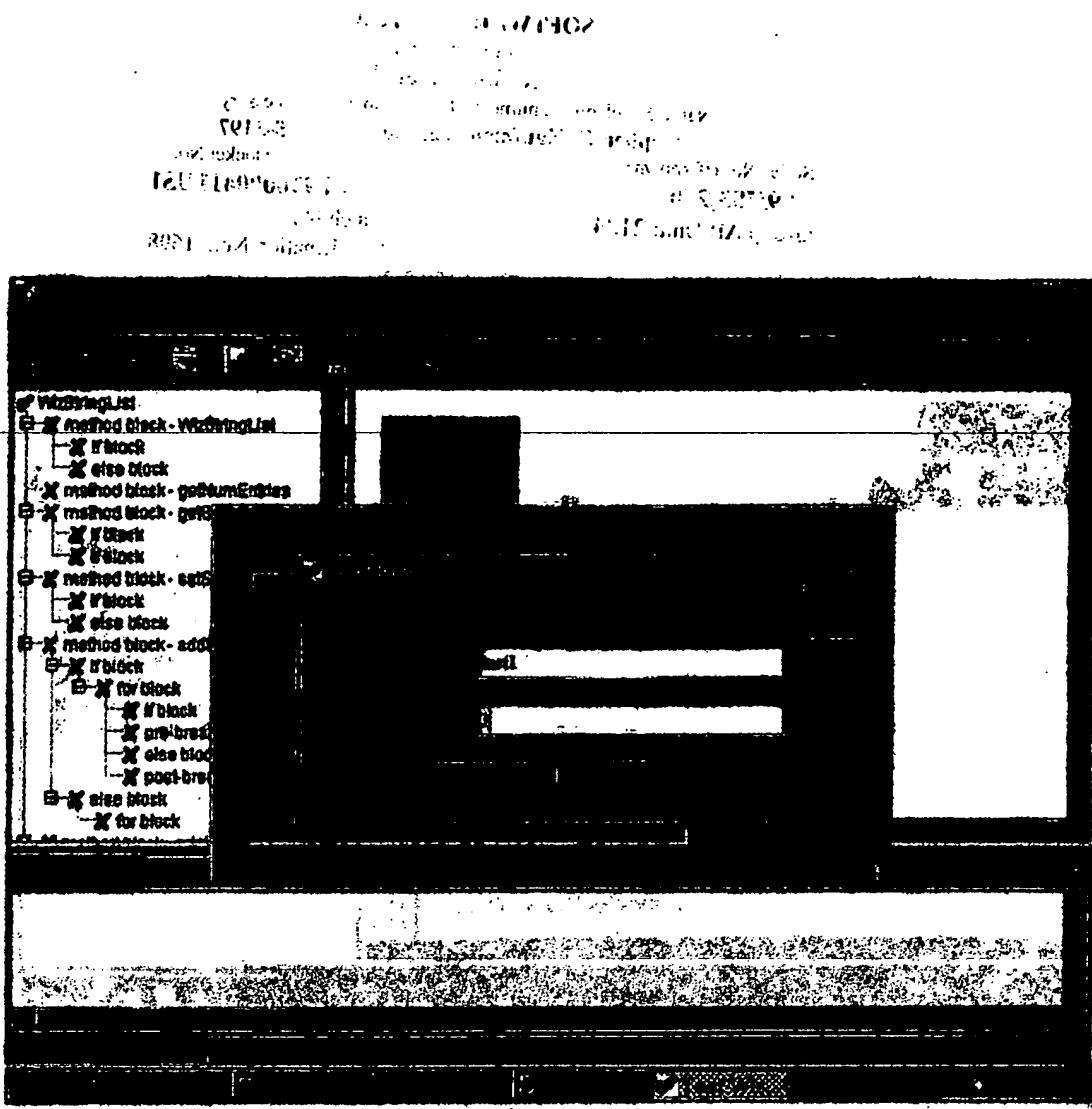


Fig. 7C

Fig. 8

**Fig. 8**

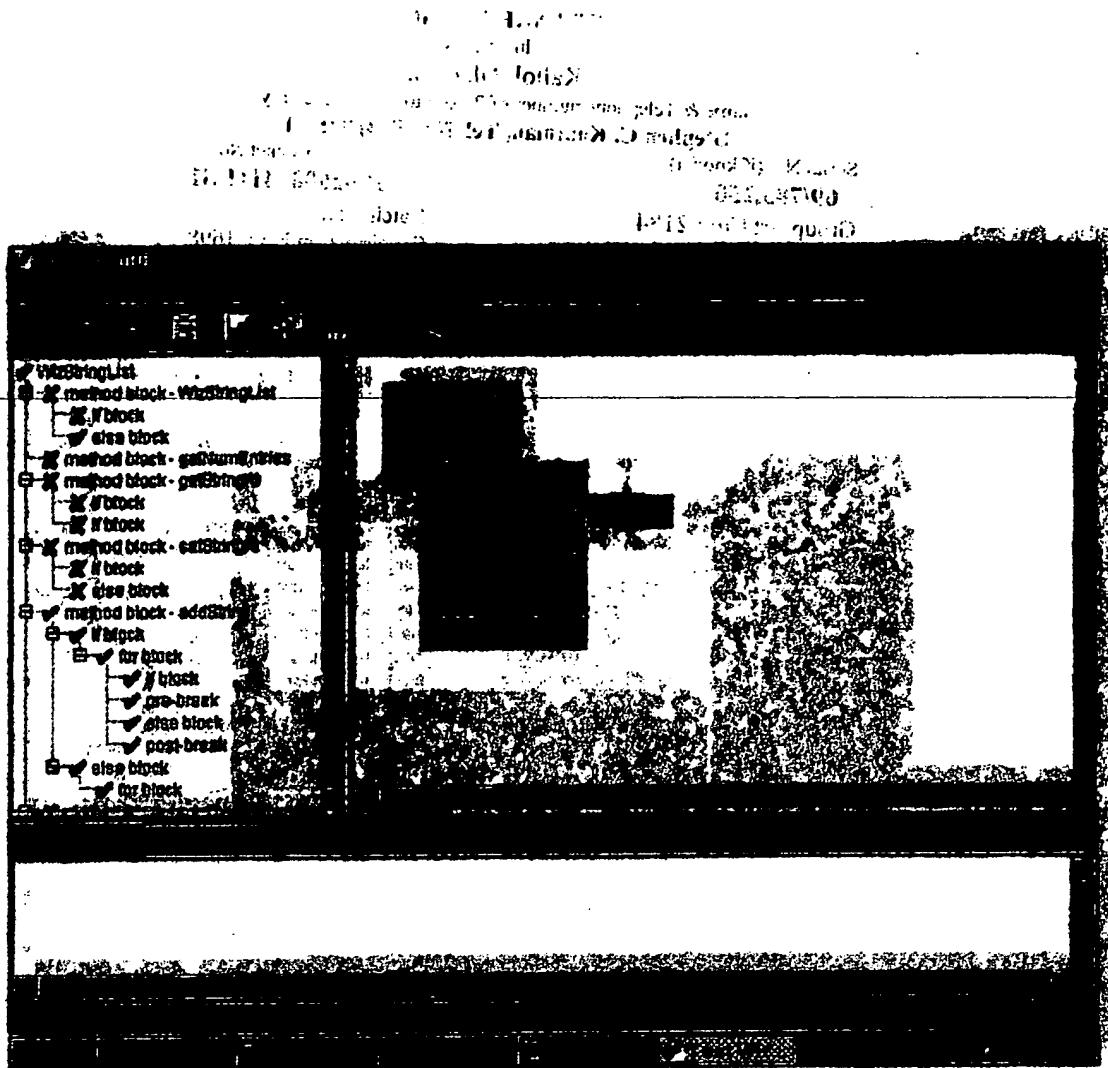
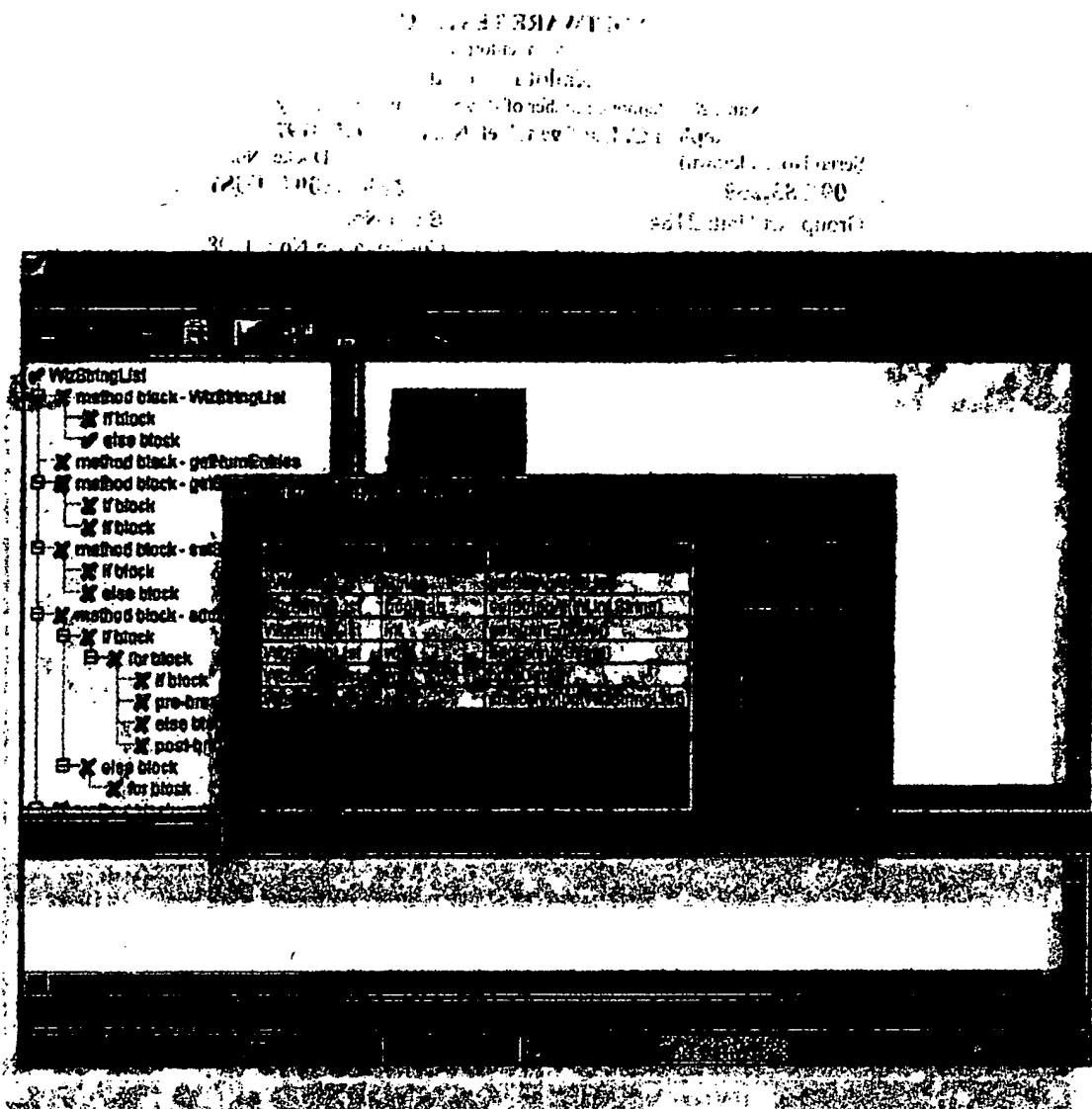


Fig. 9A



**Fig. 9B**

**Fig. 9C**

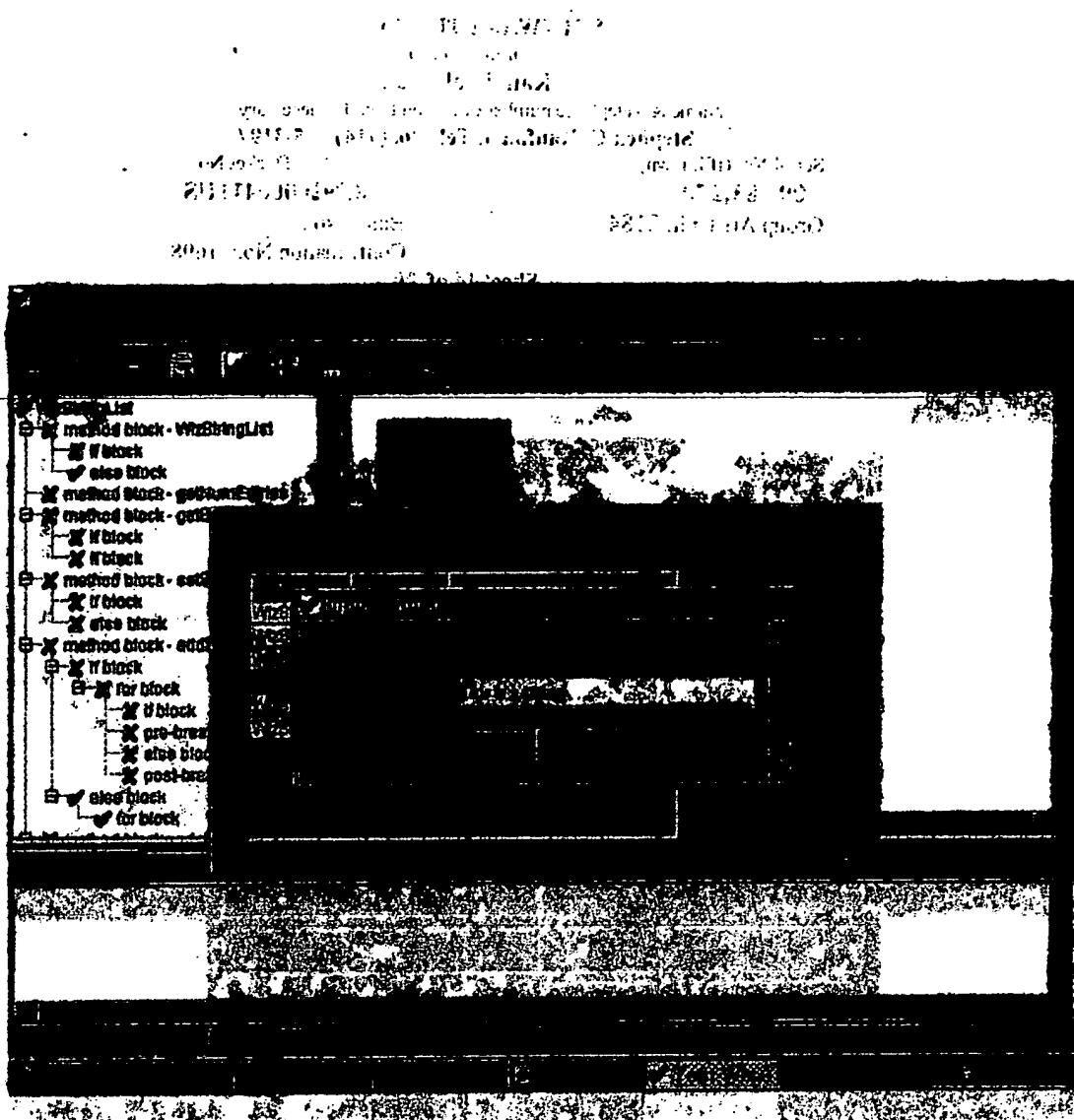


Fig. 9C

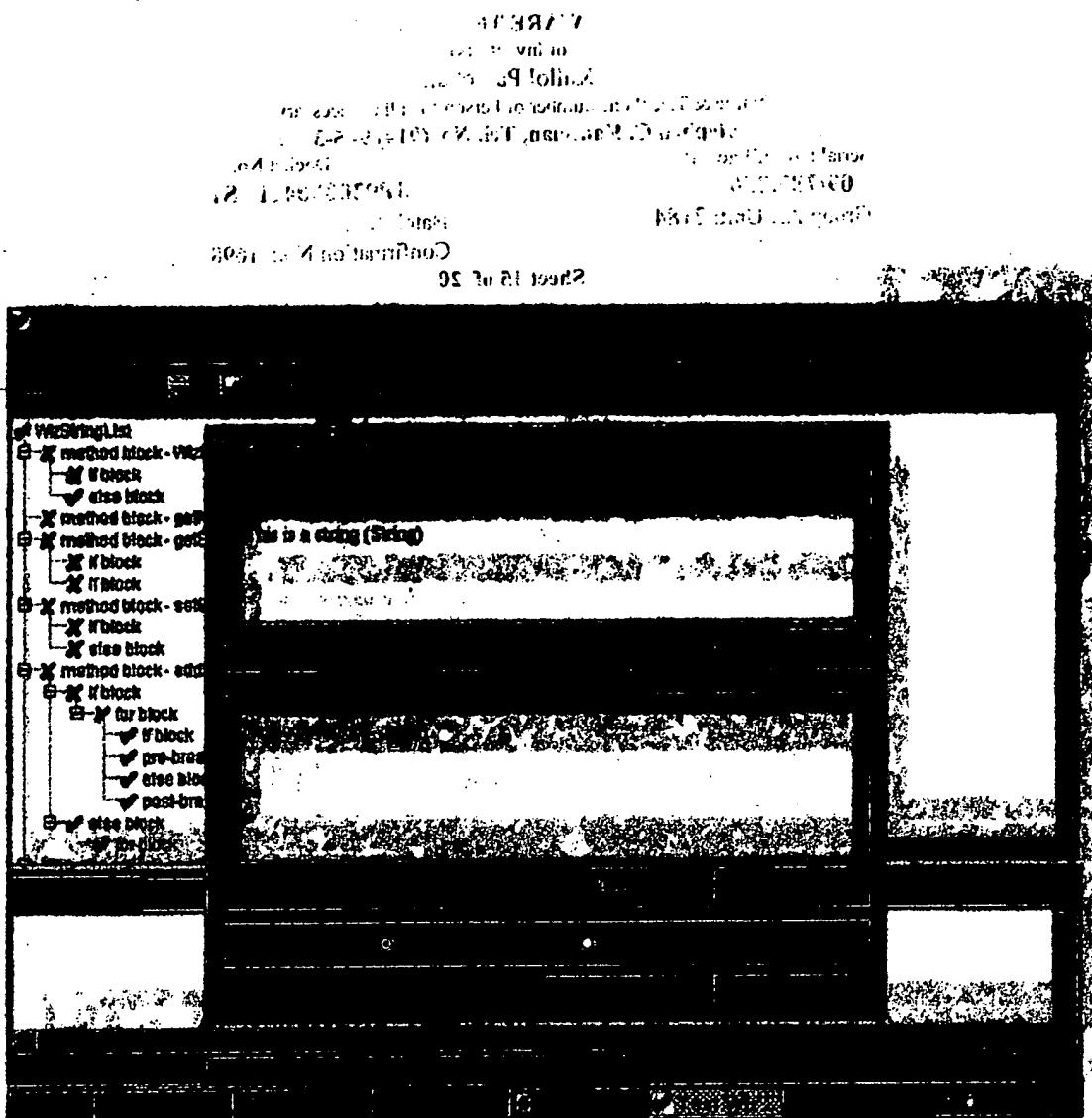


Fig. 9C  
**Fig. 9D**

17/20  
K. Pai et al.  
S.C.K. JP920000411

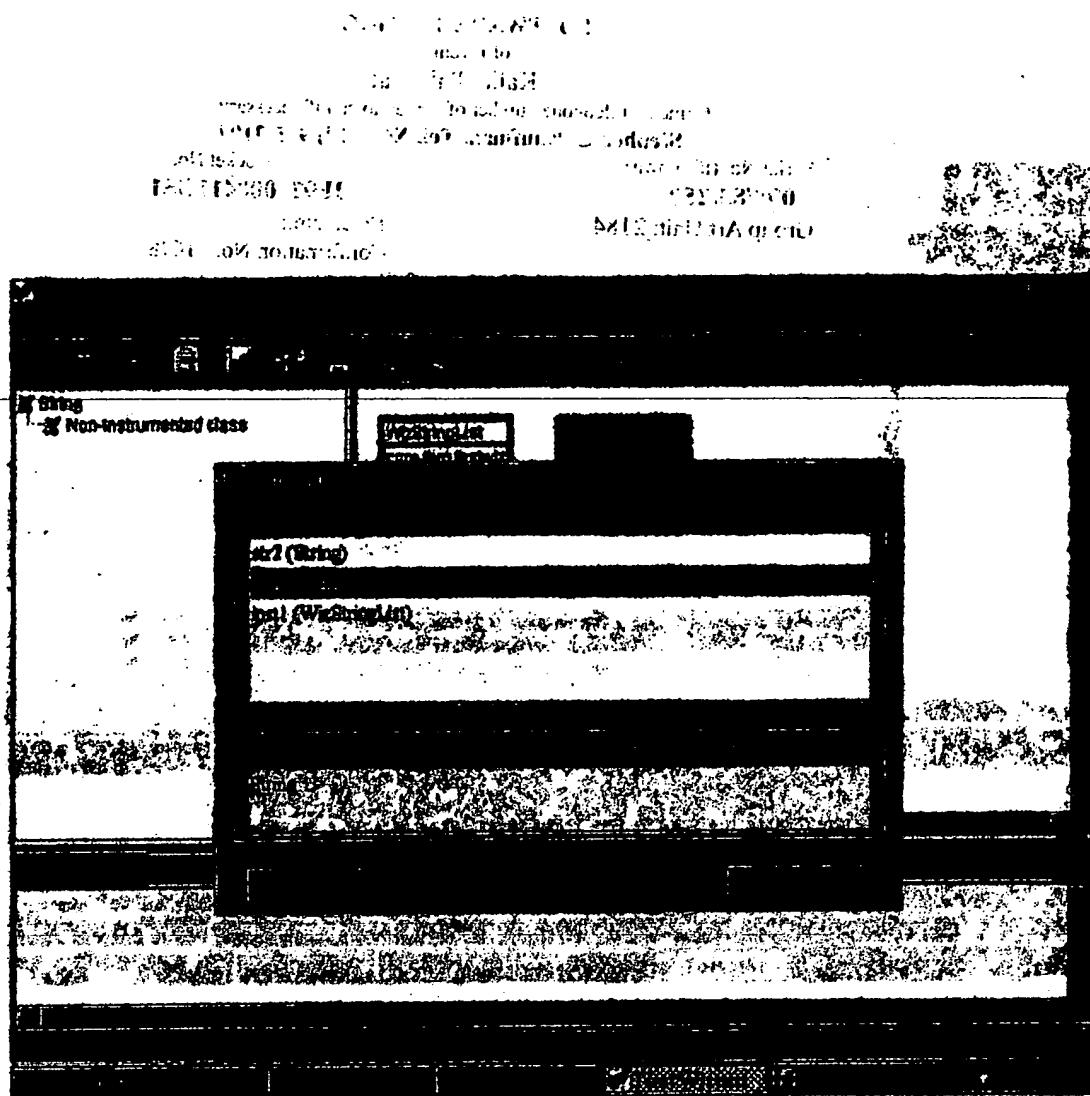


Fig. 10

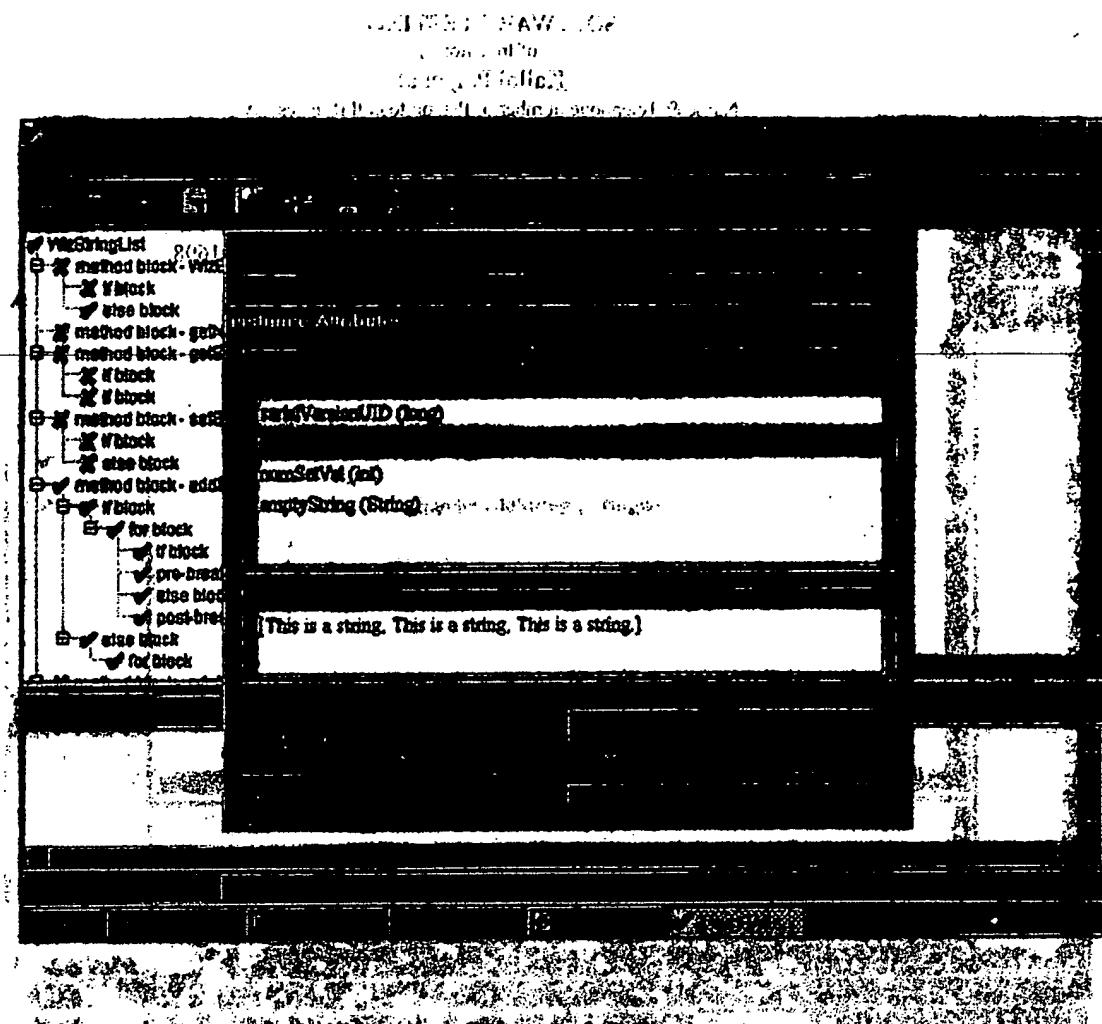


Fig. 11A

FIG. 11B

DECODED SOURCE CODE

```

class file
version 1.0
Source code
Copyright (c) 1995, Samsara and DigiSoft Inc.
All rights reserved.
File: D:\14600\src\1.java
Source code of file
class D {
    void newMacro(Vector vec) {
        System.out.println("Invocation of a macro");
        for(int index = 0; index < vec.size(); index++) {
            Vector v = (Vector)vec.get(index);
            if(v instanceof D)
                ((D)v).newMacro(vec);
        }
    }
}

((17:10:32)) W1
((17:10:32)) S1
((17:10:32)) T1
((17:10:32)) T1
((17:10:32)) T1
((17:10:32)) S1
((17:09:23)) S1

```

Fig. 11C

Fig. 11B

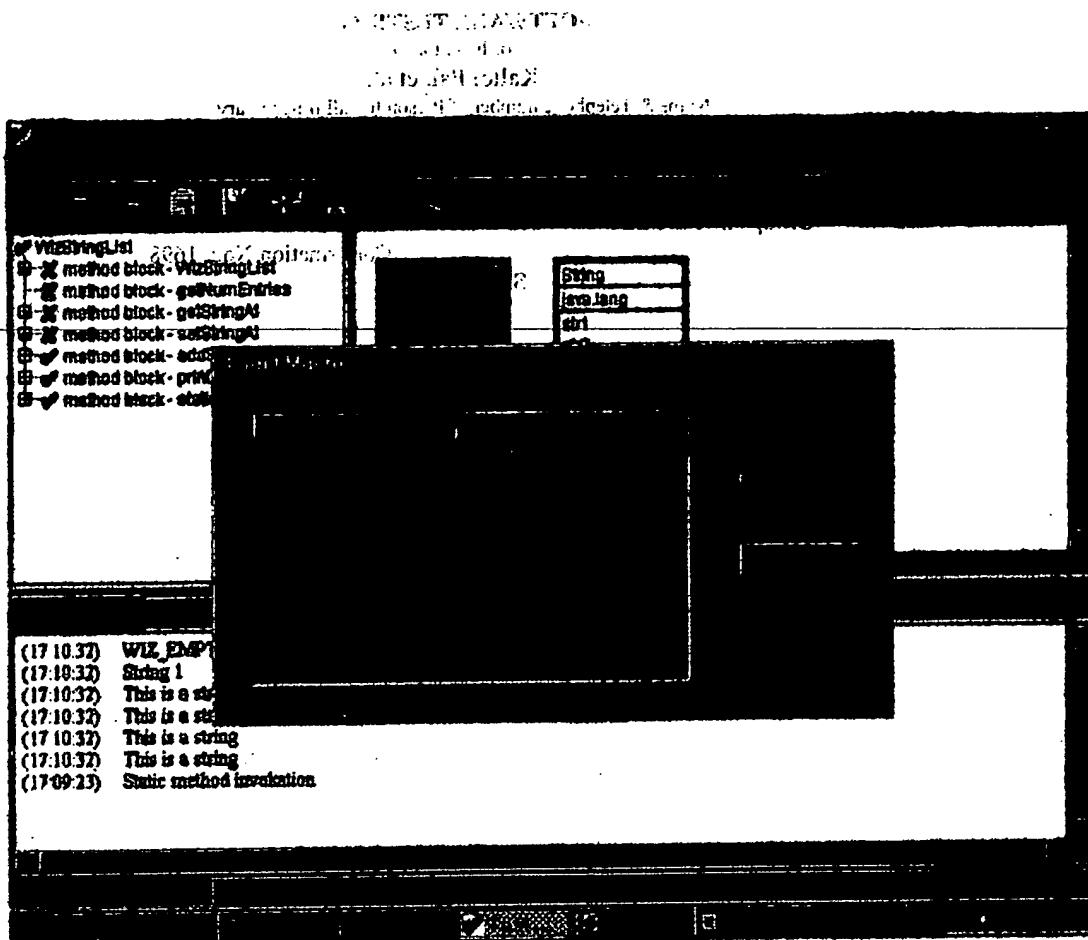


Fig. 11C